

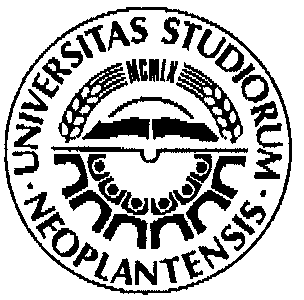
**UNIVERZITET U NOVOM SADU**

**PRIRODNO-MATEMATIČKI**

**FAKULTET**

**DEPARTMAN ZA MATEMATIKU**

**I INFORMATIKU**



Aleksandar Petrović

**OBRAZAC PPS 1**

- seminarski rad iz predmeta XML baze podataka -

Novi Sad, 2015.

**SADRŽAJ**

[1. Uvod 4](#_Toc422752886)

[2. Model 5](#_Toc422752887)

[2.1. XML šema 5](#_Toc422752888)

[2.2. Korišćene tehnologije 8](#_Toc422752889)

[2.2.1. JSF (JavaServer Faces) 8](#_Toc422752890)

[2.2.2. PrimeFaces 8](#_Toc422752891)

[2.2.3. XPath 9](#_Toc422752892)

[2.2.4. JAXB 9](#_Toc422752893)

[3. Implementacija 10](#_Toc422752894)

[3.1. Unos obrasca 10](#_Toc422752895)

[3.2. Pretraživanje obrazaca 11](#_Toc422752896)

[3.3. Izmena obrasca 12](#_Toc422752897)

[3.4. Brisanje obrasca 13](#_Toc422752898)

[3.5. Problemi 13](#_Toc422752899)

[4. Korisnički interfejs 15](#_Toc422752900)

[4.1. Početna stranica 15](#_Toc422752901)

[4.2. Novi dokument 15](#_Toc422752902)

[4.3. Pretraživanje obrazaca 17](#_Toc422752903)

[4.4. Izmena obrasca 17](#_Toc422752904)

[5. Zaključak 18](#_Toc422752905)

[6. Literatura 19](#_Toc422752906)

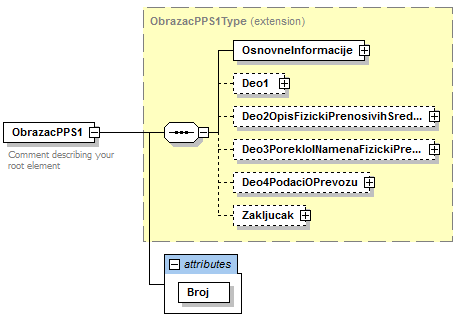
1. Uvod

Zadatak je da se napravi aplikacija za rad sa odabranim obrascem. Potrebno je kreirati odgovarajuću XML (eXstensible Markup Language) šemi za odabrani obrazac i u odnosu na tu šemu se kreiraju XML dokumenti koji će se popunjavati podacima. Popunjene XML dokumente potrebno je skladištiti na Exist serveru. Aplikacija treba da omogući popunjavanje novog obrasca podacima, pretraživanje obrazaca, izmenu postojećih i brisanje. Obrazac koji sam odabrao je “Obrazac PPS 1” koji koristi carinik na graničnom prelazu za prenos prenosivih sredstava preko državne granice Repubilike Srbije.

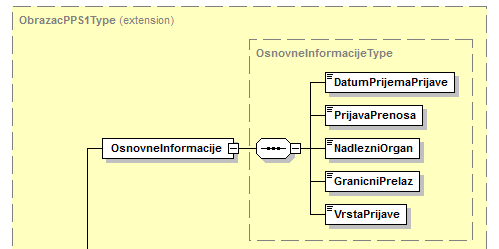
U daljem radu opisan je model kojim je predstavljen obrazac i korišćene tehnologije za kreiranje aplikacije. Zatim implementacija pojedinih delova koda i na kraju opis rada aplikacije.

1. Model
   1. XML šema

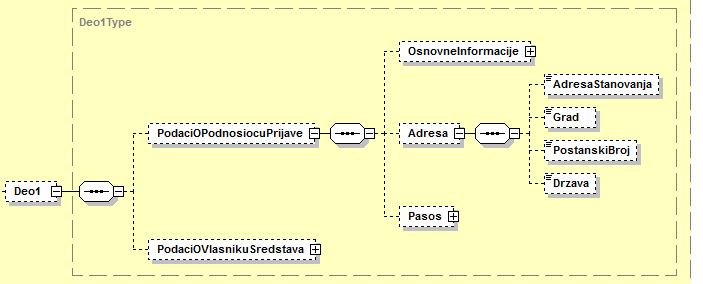
XML šema je kreirana u programu Altova XMLSpy koji je namenjen za modeliranje, transformaciju i traženje grešaka XML relacionih tehnologija. Kreirana šema omogućava popunjavanje svih podataka iz obrasca PPS 1. Obavezno je popunjavanje podataka iz poglavlja *OsnovneInformacije* (slika 2.2.) dok su ostala poglavlja opciona. To je odrađeno tako zbog pretraživanja već postojećih dokumenata, jer se oni pretražuju u odnosu na podatke iz tog poglavlja. XML šema generisana ovim dijagramom je obimna pa je neću navoditi. Tipovi podataka korišćeni u dijagramu su: String, int, float, date. Korišćen je i nabroijivi tip podataka (enum) kao na primer *VrstaPrijave* (slika 2.2.) za koju se koriste dve vrednosti, Izlazak iz RS i Ulazak u RS. Na slikama ispod prikazan je UML dijagram XML šeme obrasca PPS 1.



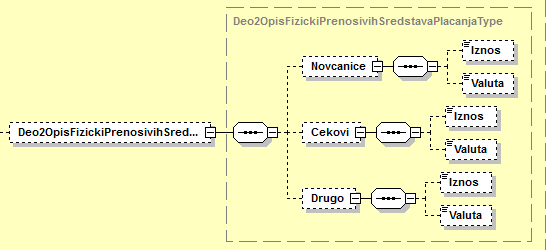
**Slika 2.1.** UML dijagram – Obrazac PPS 1



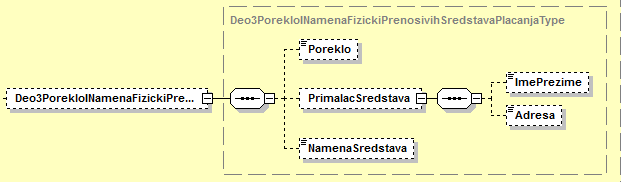
**Slika 2.2.** UML dijagram OsnovneInformacije – Obrazac PPS 1



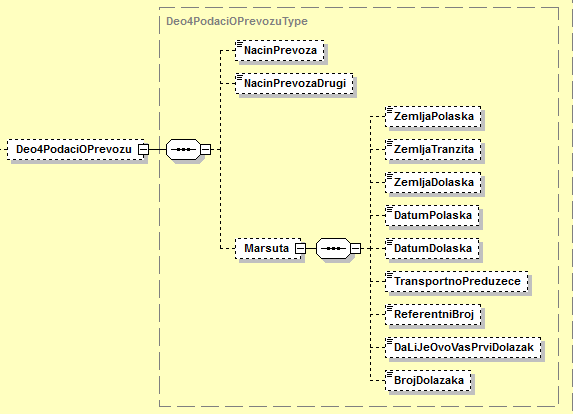
**Slika 2.3.** UML dijagram Deo1 – Obrazac PPS 1



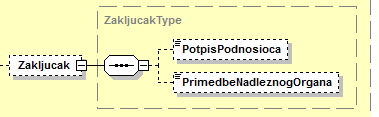
**Slika 2.4.** UML dijagram Deo2OpisFizickiPrenosivihSredstavaPlacanja – Obrazac PPS 1



**Slika 2.5.** UML dijagram Deo3PorekloINamenaFizickiPrenosivihSredstavaPlacanja – Obrazac PPS 1



**Slika 2.6.** UML dijagram Deo4PodaciOPrevozu – Obrazac PPS 1



**Slika 2.7.** UML dijagram Zakljucak – Obrazac PPS 1

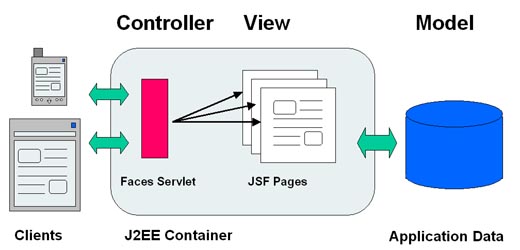
Program omogućava korisniku da kreira novi obrazac popunjavajući odgovarajuća polja koja popunjavaju XML dokument. Tako kreiran XML dokument se nakon toga pakuje i skladišti na Exist serveru.

* 1. Korišćene tehnologije
     1. JSF (JavaServer Faces)

JSF tehnologija je server-side framework za kreiranje java baziranih web aplikacija. JSF je standardizovana tehnologija za kreiranje bogatih client-server aplikacija. Ova tehnologija je bazirana na MVC ([Model–view–controller](http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller)) paternu.

Sastoji se od dva dela:

* API za reprezentaciju grafičkih komponenti i za upravljanje njihovim stanjima. Upravljanje događajima, validacija podataka na server-strani, i pretvaranje podataka. Definisanje navigacije između strana, internacionalizacija aplikacije...
* JavaServer Pages (JSP) biblioteka tagova za kreiranje grafičkih komponenti i povezivanje istih za objekte na server strani



**Slika 2.8.** JavaServer Faces arhitektura

Glavna prednost JSF tehnologije je što nudi kreiranje client-server arhitekture. Razdvajanje se postiže kreiranjem JSP stranica i takozvanih back-bean klasa.

Postoji mnogo kvalitetnih biblioteka grafičkih komponenti koje su besplatne za korišćenje i koje nam olakšavaju rad kao što su: PrimeFaces, Richfaces, ICEFaces, MyFaces…

* + 1. PrimeFaces

PrimeFaces je ekstenzija JSF tehnologije kreirana od strane [PrimeTek](http://www.primetek.com.tr/) kompanije. Pruža bogati skup grafičkih komponenti kao što su Schedule, Dialog, AutoComplete, Charts i mnoge druge.

To je Ajax bazirana tehnologija što nam omogućava da kreiramo dinamične stranice.

Veoma dobra online podrška na: <http://www.primefaces.org/showcase/index.xhtml>



* + 1. XPath

XPath (XML Path Language) je upitni jezik za selektovanje čvora ili skupa čvorova XML dokumenta. XPath je nastao od strane World Wide Web Consortium (W3C).

XPath jezik je baziran na reprezentaciji stabla XML dokumenta i omogućava kretanje kroz stablo u odnosu na sadržaj čvorova.

* + 1. JAXB

JAXB (Java Architecture for XML Binding) omogućava java programerima da mapiraju Java klase na XML šemu. JAXB pruža dve mogućnosti:

* *marshal*, tj da odgovorajuće Java klase pretvori u XML dokument
* *unmarshal*, da XML dokument pretvori u odgovarajuće Java klase.

1. Implementacija
   1. Unos obrasca

Korisnik popunjava polja na stranici *newDoc*.*xhtml* i kada klikne na dugme “Сачувај” poziva se metod *submit* klase SaveManagedBean (listing 3.2.) koji kreira novi XML dokument i skladišti na Exist serveru.

Grafički dizajn stranice je kreiran uz pomoć PrimeFaces komponenti.

<p:panelGrid styleClass=*"panelGrid"*>

<p:row>

<p:column styleClass=*"columnW20C"*>Датум рођења</p:column>

<p:column styleClass=*"columnW30C"*>

<p:calendar pattern=*"dd MMM yyyy"* locale=*"sr"* size=*"11"*

effect=*"explode"* placeholder=*"Изабери датум"*

value=*"#{saveManagedBean.model.datumRodjenjaP}"* navigator=*"true"*

yearRange=*"c-100:c+1"*>

</p:calendar>

</p:column>

<p:column styleClass=*"columnW20C"*>Датум рођења</p:column>

<p:column styleClass=*"columnW30C"*>

<p:calendar pattern=*"dd MMM yyyy"* locale=*"sr"* size=*"11"*

effect=*"explode"* placeholder=*"Изабери датум"*

value=*"#{saveManagedBean.model.datumRodjenjaV}"* navigator=*"true"*

yearRange=*"c-100:c+1"*>

</p:calendar>

</p:column>

</p:row>

</p:panelGrid>

**Listing 3.1.** Deo implementacije koda *newDoc.xhtml* stranice za kreiranje novog dokumenta

Naveden kod (listing 3.1.) predstavlja kreiranje jednog reda obrasca na stranici *newDoc*.*xhtml*. Najvažniji deo ovog koda je atribut *value* koji predstavlja vezivanje polja unos teksta ili u ovom slučaju datuma za objekat server klase *SaveManagedBean*. *Value* je predstavljen EL (Expression Language) sintaksom.

Static metoda *saveDocument* klase *DatabaseAccess* dobija kao parameter objekat klase *MyModel* koji u sebi sadrži sva polja kao i Obrazac PPS 1 i nad tim objektom odradi JAXB marshal i sačuva kreirani XML dokument u bazi.

@ManagedBean

@RequestScoped

**public** **class** SaveManagedBean {

MyModel mymodel;

@PostConstruct

**private** **void** init() {

mymodel = **new** MyModel();

}

**public** **void** submit() {

DatabaseAccess.*saveDocument*(mymodel);

}

**public** MyModel getModel() {

**return** mymodel;

}

**public** **void** setModel(MyModel mymodel) {

**this**.mymodel = mymodel;

}

}

**Listing 3.2.** Back-bean klasa za kreiranje novog dokumenta

* 1. Pretraživanje obrazaca

Mogu se pretraživati postojeći dokumenti u odnosu na elemente sa slike 2.2.

Za pretraživanje se koristi XPath, koji u odnosu na popunjenost polja na stranici *search.xhtml* pretražuje određene čvorove XML dokumenta. Kada se izvrši upit na Exist serveru dobiju se svi dokumenti iz baze koji odgovaraju izgenerisanom XPath upitu, oni se unmarshal-uju i kreira se instanca klase *MyModel* za svaki dokument koja se popunjava poljima prethodno kreiranih objekata metodom *unmarshal* (listing 3.3).

Nakon izvršavanja XPath upita korisnik dobija sve dokumente koji odgovaraju upitu i prikazuju mu se u tabeli na stranici *searchDoc.xhtml,* što se može videti u sledećem poglavlju.

**public** **static** List<MyModel> getAllDocumentsByCriteria(String broj, Date datumPrijema, String nadlezniOrgan, String granicniPrelaz, String vrstaPrijave) {

QueryManager qm = **new** QueryManager(ExistManager.***COLLECTION***);

String query = "/ObrazacPPS1";

query += "[";

**if** (!broj.equals(""))

query += "@Broj='" + broj + "'";

**if** (datumPrijema != **null**) {

**if** (!(query.charAt(query.length() - 1) == '['))

query += " and ";

query += "OsnovneInformacije/DatumPrijemaPrijave/text()='" + *getXmlGregorianCalendarFromDate*(datumPrijema).toString().replace("T00:00:00.000", "") + "'";

}

**if** (!nadlezniOrgan.equals("")) {

**if** (!(query.charAt(query.length() - 1) == '['))

query += " and ";

query += "OsnovneInformacije/NadlezniOrgan/text()='" + nadlezniOrgan + "'";

}

**if** (!granicniPrelaz.equals("")) {

**if** (!(query.charAt(query.length() - 1) == '['))

query += " and ";

query += "OsnovneInformacije/GranicniPrelaz/text()='" + granicniPrelaz + "'";

}

**if** (vrstaPrijave != **null**) {

**if** (!(query.charAt(query.length() - 1) == '['))

query += " and ";

query += "OsnovneInformacije/VrstaPrijave/text()='" + vrstaPrijave + "'";

}

query += "]";

**if** (query.contains("[]")) {

query = query.replace("[]", "");

}

List<XMLResource> res = qm.performXpath(query);

List<MyModel> models = **new** LinkedList<MyModel>();

**try** {

**for** (**int** i = 0; i < res.size(); i++) {

Node doc = res.get(i).getContentAsDOM();

JAXBContext jc = JAXBContext.*newInstance*("model");

Unmarshaller u = jc.createUnmarshaller();

JAXBElement<ObrazacPPS1> el = u.unmarshal(doc, ObrazacPPS1.**class**);

ObrazacPPS1 obrazac = el.getValue();

models.add(*makeModel*(obrazac, res.get(i).getDocumentId()));

}

} **catch** (Exception e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

**return** models;

}

**Listing 3.3.** Metod klase *DatabaseAccess* za pretraživanje XML dokumenata u bazi

u odnosu na zadate parametre

* 1. Izmena obrasca

Nakon što korisnik pretraži dokumente on može da izabere željeni dokument iz tabele i da izvrši izmene nad istim.

Implementacija dela koda za izmenu je slična kao i pri kreiranju novog dokumenta samo što se prvo obriše taj dokument iz baze i sačuva izmenjeni kao novi dokument.

**public** **static** **void** changeDocument(MyModel myModel) {

ExistManager.*remove*(myModel.getDocumentId());

*saveDocument*(myModel);

}

**Listing 3.4.** Metod klase *DatabaseAccess* za izmenu dokumenta

**public** **static** **final** String ***COLLECTION*** = "/db/AleksandarPetrovic";

**public** **static** **final** String ***URI*** = DBConnection.*URI* + "/db/AleksandarPetrovic";

**public** **static** **void** remove(String documentId) {

**try** {

Collection col = DatabaseManager.*getCollection*(***URI***,"admin","xmlbp");

Resource res = col.getResource(documentId);

col.removeResource(res);

} **catch** (XMLDBException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

**Listing 3.5.** Metod klase *ExistManager* za brisanje dokumenta iz baze

* 1. Brisanje obrasca

Listing 3.5. prikazuje implementaciju koda za brisanje dokumenta sa zadatim imenom dokumenta u bazi.

* 1. Problemi
* Jedan od problema sa kojima sam se susreo je kada sam pokušao da odradim celu web aplikaciju na ćirilici. JavaScript fajl calendar.js koji PrimeFaces komponenti calendar dodeljuje lokalizaciju na ćirilici se mora pretvoriti iz formata ANSI u format UTF-8.
* Kada JAXB generiše klase iz XML šeme podatak tipa datum je predstavljen kao tip klase *XMLGregorianCalendar*. Kako PrimeFaces komponenta calendar prihvata za vrednost *java.util.Date* onda se vrši konverzija iz *XMLGregorianCalendar* u *Date* i iz *Date* u *XMLGregorianCalendar* u zavisnosti od primene (listing 3.6, 3.7).

**private** **static** Date xmlGregorianCalendartoDate(XMLGregorianCalendar calendar) {

**if** (calendar == **null**) {

**return** **null**;

}

return calendar.toGregorianCalendar().getTime();

}

**Listing 3.6.** Metod klase *DatabaseAccess* za konverziju datuma iz *XMLGregorianCalendar*

formata u java *Date*

**private** **static** XMLGregorianCalendar getXmlGregorianCalendarFromDate(Date date) {

**try** {

GregorianCalendar c = **new** GregorianCalendar();

c.setTime(date);

**return** DatatypeFactory.*newInstance*().newXMLGregorianCalendar(c);

} **catch** (DatatypeConfigurationException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

**return** **null**;

}

**Listing 3.7.** Metod klase *DatabaseAccess* za konverziju datuma iz formata java *Date* u *XMLGregorianCalendar*

1. Korisnički interfejs
   1. Početna stranica

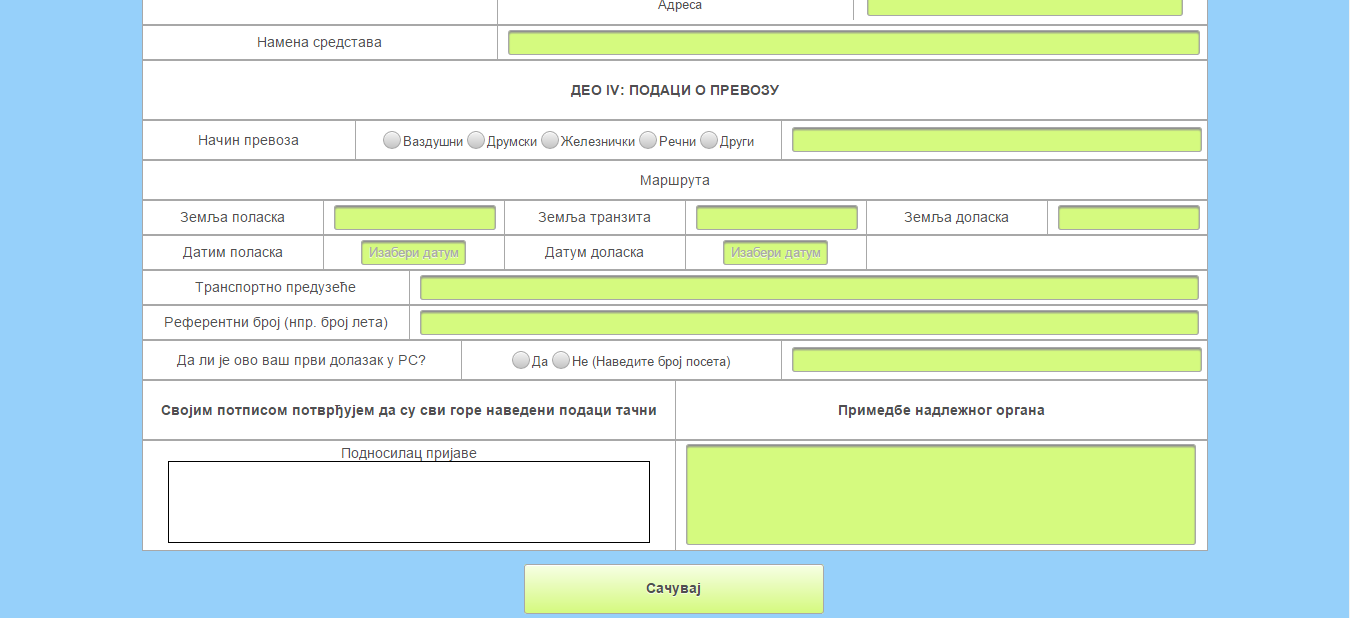
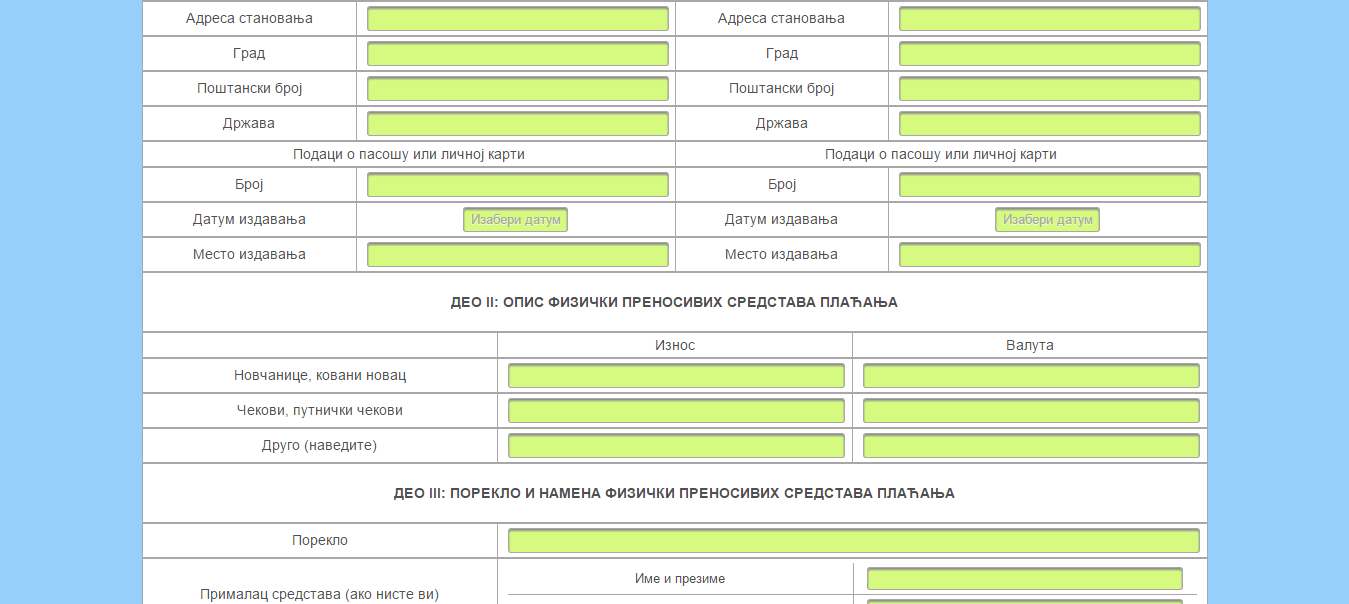
Početna stranica index.xhtml. Korisnik klikom na jedno od dva dugmeta otvara novu stranicu za kreiranje novog obrasca ili za pretraživanje postojećih obrazaca.



**Slika 4.1.** Početna stranica web aplikacije

* 1. Novi dokument

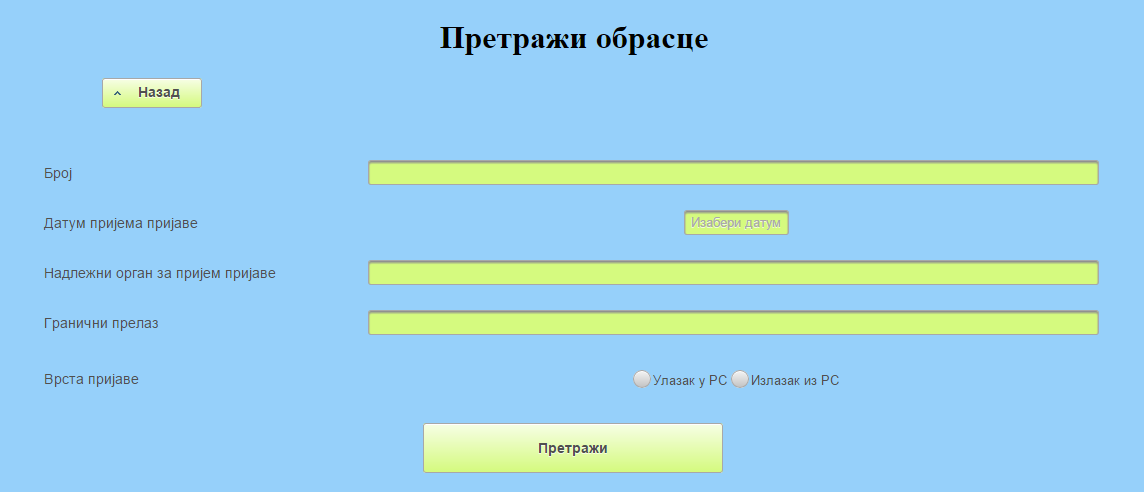
Nakon što je korisnik na početnoj stranici odabrao da želi da kreira novi dokument otvara se nova stranica (slika 4.2.). Popunjavanjem praznih polja i klikom na dugme „Сачувај“ kreira se novi document u bazi.



**Slika 4.2.** Stranica za kreiranje novog dokumenta

* 1. Pretraživanje obrazaca

Korisnik popunjava polja i nakon toga klikom na dugme „Претражи“ otvara se nova stranica sa svim dokumentima koji zadovaljavaju unete parametre. Ako ne unese parametre, kao rezultat će dobiti sve dokumente iz baze (slika 4.4.).



**Slika 4.3.** Stranica za unos parametara pretrage



**Slika 4.4.** Primer pretraživanja obrazaca – svi obrasci iz baze

* 1. Izmena obrasca

Obrazac koji želimo da izmenimo odaberemo klikom na dugme „Измени“ (slika 4.4). Nakon klika otvara nam se stranica *showDoc*.*xhtml* (slika 4.2.) sa bitnom razlikom da su polja već popunjena obrascem iz baze i da ih korisnik može izmeniti i sačuvati nove izmene.

1. Zaključak

Korišćenje XML native baze podataka za skladištenje dokumenata pruža niz prednosti i mana. Potrebe ovog zadatka su zahtevale XML native baze podataka s tim što je ovo moglo da se uradi i sa relacionom bazom podataka ili sa skladištenjem u fajlu. Iskorišćenost relacionih baza u odnosu na XML native baza je neuporedivo veće. Isto tako i veoma jednostavnim upitima XPath uputnog jezika možemo dobiti željene dokumente.

U sledećoj verziji programa se treba asocirati na dodavanje novih obrazaca.

1. Literatura
2. Goncalves, A. (2010). *Beginning Java EE 6 with GlassFish 3.*
3. *JavaServer Faces (JSF) Tutorial*. (2015, Jun 21). Preuzeto sa Tutorials point: http://www.tutorialspoint.com/jsf/
4. *JavaServer Faces*. (2015, Jun 21). Preuzeto sa Wikipedia: https://en.wikipedia.org/wiki/JavaServer\_Faces
5. *JavaServer Faces Technology*. (2015, Jun 21). Preuzeto sa The Java EE 6 Tutorial: http://docs.oracle.com/javaee/6/tutorial/doc/bnaph.html
6. Škrbić, S. (2012). *XML native baze podataka.* Novi Sad.